CBC Database Architecture Solution

Uladzimir Kasacheuski, NN:18

**Executive Summary**

The CBC project scope included the several requirements important to a self-aware and efficient company. Concretely, it included keeping current and historical data records for several entities, supporting the communication of time sensitive information, and employee management. Business rules and real data were steadily shared with the design team to enable the creation of a solution which would support and optimize the functionality of the required by the CBC database architecture.

The solution presented in this report utilizes a relational database model to ensure availability and consistency of data. The database architecture was designed using both top down and bottom up techniques to eliminate update anomalies, ensuring accurate results in performance. Included in this report are the results generated by a functional prototype of the design solution, demonstrating its fulfillment of CBC project requirements.

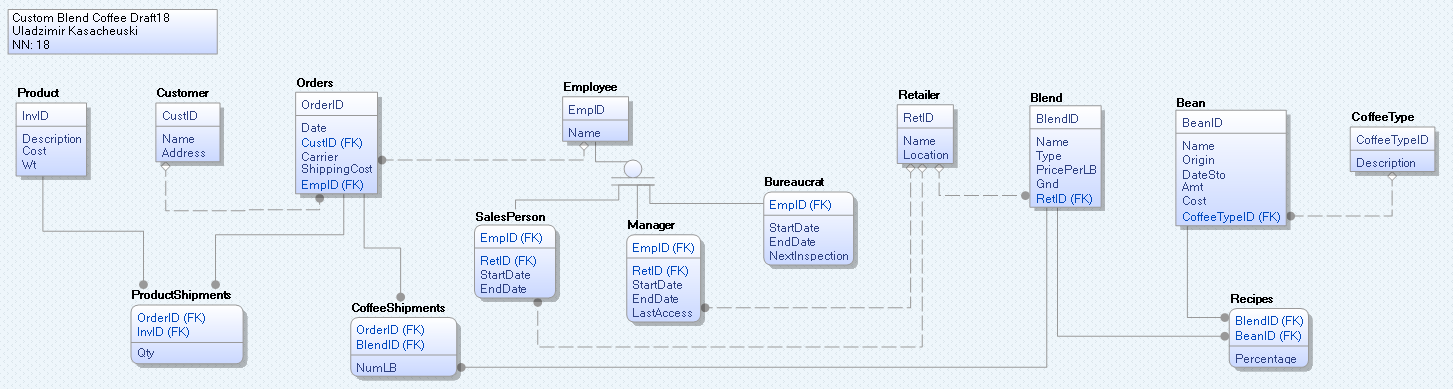
**Solution**

The database architecture presented in this report is built according to a relational database model. The reason a relational model is the optimal candidate for this project is conveyed below. The ultimate design, iterated and optimized as information was delivered, is presented after that both graphically and with a brief explanation. Historical progression of the database design can be found in the appendix (A1). The methods employed to generate the solution can be found in the ‘Methods’ section of the report.

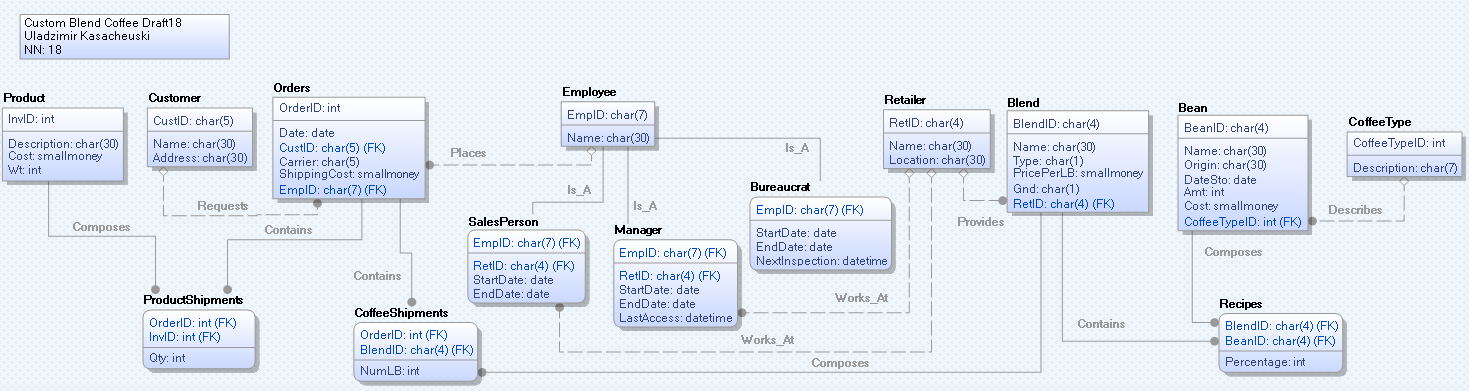
The relational database model is the optimal candidate for the CBC project requirements. A relational database ensures data consistency and availability with full ACID support. For tracking customer purchases, this is critical, as any data loss or downtime can cost a business a lot of money. In addition to important features baked into the theoretical foundations of the relational model, relational databases are supported advanced security features, concurrency features, and more, offered by industry leading software such as SQL Server.

The Logical and Physical data models are presented on the next two pages with a brief explanation of the features of the solution.

Logical Data Model



Physical Data Model



A brief description of the design modeled above follows. Details as to the methods and rationale employed in developing this database design can be found in the ‘Methods’ section.

The database design presented encapsulates the all entities required to be tracked, supports all the functional requirements stated in the CBC project scope, and ensures data consistency at the design level by achieving the maximum normalization level possible, 5NF.

The four main entities required for functionality and data tracking requested in the scope are orders (including customer information), blends (including their constituents), project personnel, and retail locations. All are distinctly identifiable in the data models presented above. All other entities present are either supporting entities (such as beans - without which you cannot have blends) or entities which are products of ensuring data consistency at the logical level (products of normalization).

Attribute domains were chosen to minimize data usage based on the data provided. Structured attributes, such as ID’s, were particularly limited as the minimal required size can be pinpointed due to their low size variance nature. Other attributes, like names and descriptions, required a greater range to support their highly variable sizes.

Care was taken in not overstepping the scope defined by the project - ensuring a timely delivery and a low cost solution for the client. Any additional or extension of functionality that is required can be discussed and supported by our team.

**Results**

To demonstrate the successful fulfillment of the presented database design in regards to the CBC project requirements, a few results from typical and more advanced database usage will be displayed. These results were generated on a prototype database, constructed from the relational database design presented in our solution. The prototype was constructed on the SQL Server platform by using the Forward Generation feature of the ERwin software. The DDL used in the process is included in the appendix (A2). The database was loaded with all data provided by CBC in this project. The DML is also included in the appendix (A3).

The results are structured in three parts: an English statement of what is desired, the SQL query that produces the results, and the results generated by the query.

*Recording New Information*

1) Add an employee as a sales person for the New York store.

INSERT INTO Employee (EmpID, Name) VALUES ('EMP918', 'Uladzimir Kasacheuski')

(1 row(s) affected)

INSERT INTO SalesPerson (EmpID, RetID) VALUES ('EMP918', 'NYC1')

(1 row(s) affected)

2) Create blend with BlendID ‘N03’, named ‘Spec03’ which is 20% Kona coffee and 80% Brazil Dark

INSERT INTO Blend (BlendID, Name, Type, PricePerLB, Gnd, RetID)

VALUES ('N03', 'Spec03', 'S', 30.90, 'G', 'NYC1');

*INSERT INTO Recipes (BlendID, BeanID, Percentage) VALUES ('N03', 'K001', 20);*

*INSERT INTO Recipes (BlendID, BeanID, Percentage) VALUES ('N03', 'B234', 80);*

(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)

3) Create an order (Ord# 5000) that you sell to Bill Green consisting of one pound of ground coffee for the new blend, ‘N03’,

INSERT INTO Orders (OrderID, Date, CustID, Carrier, ShippingCost, EmpID) VALUES (5000, '2016-12-09', 'CC201', 'UPS', 12.00, 'EMP918');

INSERT INTO CoffeeShipments (OrderID, BlendID, NumLB) VALUES (5000, 'N03', 1);

(1 row(s) affected)

(1 row(s) affected)

*Questions*

1) What is the largest cost for a pound of standard beans?

SELECT MAX(Cost) AS 'Maximum Cost Per Pound of Standard Beans' FROM Bean;

Maximum Cost Per Pound of Standard Beans

----------------------------------------

81.50

(1 row(s) affected)

2) For each type of standard bean, list in chronological (Date) order the orders that contain that type of bean.

SELECT Bean.BeanID, Orders.Date, Orders.OrderID, Blend.BlendID, Recipes.Percentage,

CoffeeShipments.NumLB

FROM Bean, Blend, Recipes, CoffeeShipments, Orders

WHERE Bean.BeanID = Recipes.BeanID AND Recipes.BlendID = Blend.BlendID

AND CoffeeShipments.BlendID = Blend.BlendID AND Orders.OrderID = CoffeeShipments.OrderID

ORDER BY Bean.BeanID ASC, Orders.Date ASC;

BeanID Date OrderID BlendID Percentage NumLB

------ ---------- ----------- ------- ----------- -----------

B234 2016-07-20 1015 CS03 50 1

B234 2016-07-21 2000 N02 90 1

B234 2016-08-01 2050 N02 90 2

B234 2016-08-02 2051 CS05 75 5

B234 2016-08-10 3015 CS04 60 1

B234 2016-09-02 4000 CS05 75 4

B234 2016-12-09 5000 N03 80 1

CO23 2016-08-02 2051 CS05 25 5

CO23 2016-08-10 3016 SFOA 50 1

CO23 2016-09-02 4000 CS05 25 4

CR05 2016-08-10 3015 CS04 20 1

K001 2016-06-15 1012 CR01 100 1

K001 2016-07-20 1015 CS03 50 1

K001 2016-07-21 2000 N02 10 1

K001 2016-08-01 2050 N02 10 2

K001 2016-09-02 4000 CR01 100 2

K001 2016-12-09 5000 N03 20 1

K002 2016-06-15 1013 CR02 100 1

P001 2016-08-10 3015 CS04 20 1

P001 2016-08-10 3016 SFOA 50 1

P002 2016-08-10 3017 SFOP 100 4

P002 2016-08-10 2055 SFOP 100 5

(22 row(s) affected)

3) What blends does the New York 1 store make available for purchase?

SELECT Blend.BlendID, Blend.Name From Blend WHERE Blend.RetID = 'NYC1'

BlendID Name

------- ------------------------------

N02 Spec02

N03 Spec03

(2 row(s) affected)

4) List the standard beans and the amount on hand in stock in chronological order by the date they were placed in storage.

SELECT Bean.BeanID, Bean.Name, Bean.Amt, Bean.DateSto From Bean ORDER BY Bean.DateSto ASC

BeanID Name Amt DateSto

------ ------------------------------ ----------- ----------

CR05 Tarrazu 33 2015-01-01

B234 Brazil Dark 225 2016-04-01

K001 Kona #1 225 2016-05-01

K002 Kona #2 110 2016-05-01

CO23 Supremo 50 2016-06-01

P001 High Grown 100 2016-07-01

P002 High Grown 100 2016-07-01

(7 row(s) affected)

5) List the orders that were processed in June 2016 or August 2016.

SELECT Orders.OrderID, Orders.Date

FROM Orders

WHERE (MONTH(Date) = '06' OR MONTH(Date) = '08') AND YEAR(Date) = '2016'

OrderID Date

----------- ----------

1012 2016-06-15

1013 2016-06-15

2050 2016-08-01

2051 2016-08-02

2055 2016-08-10

3015 2016-08-10

3016 2016-08-10

3017 2016-08-10

(8 row(s) affected)

6) What blends contain Kona coffee?

SELECT DISTINCT Blend.BlendID, Blend.Name

FROM Bean, Blend, Recipes

WHERE Bean.BeanID = Recipes.BeanID AND Blend.BlendID = Recipes.BlendID AND Bean.Origin LIKE '%Kona%';

BlendID Name

------- ------------------------------

CR01 Kona #1

CR02 Kona #2

CS03 Dark

N02 Spec02

N03 Spec03

(5 row(s) affected)

7) What blends were ordered in orders placed by retailer New York 1?

SELECT DISTINCT Blend.BlendID, Blend.Name

FROM Blend, CoffeeShipments, Orders, SalesPerson, Retailer

WHERE Blend.BlendID = CoffeeShipments.BlendID AND Orders.OrderID = CoffeeShipments.OrderID

AND Orders.EmpID = SalesPerson.EmpID AND Retailer.RetID = SalesPerson.RetID AND

Retailer.RetID = 'NYC1';

BlendID Name

------- ------------------------------

N02 Spec02

N03 Spec03

(2 row(s) affected)

8) List the customers and the order number for those customers who have coffee cups on order.

SELECT Customer.Name, Customer.CustID, Orders.OrderID, Product.Description

FROM Product, ProductShipments, Customer, Orders

WHERE Product.Description LIKE '%cup%' AND Product.InvID = ProductShipments.InvID

AND ProductShipments.OrderID = Orders.OrderID AND Customer.CustID = Orders.CustID

CustID Name OrderID Description

------ ------------------------------ ----------- ------------------------------

AC101 Anne Green 1013 CBC cup (W)

SS205 John Doe 2050 CBC cup (W)

SS205 John Doe 2050 CBC cup (B)

(3 row(s) affected)

9) Who (Name and EMP code) manages the New York 1 store?

SELECT Employee.EmpID, Employee.Name

FROM Employee, Manager

WHERE Employee.EmpID = Manager.EmpID AND Manager.RetID = 'NYC1'

EmpID Name

------- ------------------------------

EMP0102 Mary Manager

(1 row(s) affected)

10) Who (Name and EMP code) has access to data about the New York 1 store orders?

SELECT DISTINCT Employee.EmpID, Employee.Name

FROM Employee, Manager, SalesPerson, Bureaucrat

WHERE (Employee.EmpID = Manager.EmpID AND Manager.RetID = 'NYC1')

OR (Employee.EmpID = SalesPerson.EmpID AND SalesPerson.RetID = 'NYC1')

OR (Employee.EmpID = Bureaucrat.EmpID) /\* Assuming Bureaucrats have access to all stores \*/

EmpID Name

------- ------------------------------

EMP0000 Bob Bureaucrat

EMP0102 Mary Manager

EMP109 Jane Smith

EMP202 Frank Martin

EMP918 Uladzimir Kasacheuski

(5 row(s) affected)

11) How many orders have been made by each salesperson (Name and EMP code)?

SELECT Employee.EmpID, Employee.Name, COUNT(Orders.OrderID) AS 'Number Of Orders'

FROM Employee, Orders

WHERE Employee.EmpID = Orders.EmpID GROUP BY Employee.EmpID, Employee.Name

EmpID Name Number Of Orders

------- ------------------------------ ----------------

EMP108 Bill Brown 4

EMP109 Jane Smith 1

EMP202 Frank Martin 1

EMP213 Anne Dough 2

EMP303 Mike Green 1

EMP309 Alice Grey 1

EMP400 Bill Brown 1

EMP918 Uladzimir Kasacheuski 1

(8 row(s) affected)

12) List the types of beans and percentages in blend Spec03.

SELECT Bean.BeanID, Bean.Name, Recipes.Percentage FROM Bean, Recipes WHERE Bean.BeanID = Recipes.BeanID AND Recipes.BlendID = 'N03'

BeanID Name Percentage

------ ------------------------------ -----------

B234 Brazil Dark 80

K001 Kona #1 20

(2 row(s) affected)

**Methods**

After assessing database model options and deciding on the relational database model, architecting the database design presented in this report employed the use of both Top Down and Bottom Up methodologies. After a naive logical model of the data was generated, the process of normalization was applied to eliminate update anomalies and data redundancy. This process ensures that data will remain consistent as a product of the logical design of the database.

The design started with the data, using the bottom up methodology. All universal relations provided by the CBC team for this project were subjected to functional dependency analysis. Results of these analyses can be found in the appendix (A4). The functional dependencies generated by these analyses were then interpreted and subjected to the business rules provided through the enterprise statements provided by the CBC team, employing the top down methodology.

The requirements of the CBC project included all of the following: keep track of orders consisting of one or more coffee shipments or one or more inventory shipments, keep records of which stores sold what orders, keep track of the special blends unique to each retailer, support the communication of order and customer information from retailers to fulfillment, and keep track of project personnel including last access times for managers, the date and time of a bureaucrats next inspection, and the amount of orders placed by a salesperson. All of these requirements were critically considered, in addition to other business rules, while employing the top down design methodology.

After refining the functional relationships down further with the top down approach, we began the process of normalization. The database design was thoroughly normalized to the 5NF level to ensure data consistency at the logical level, a trademark of good database design. Functional dependencies were analyzed at each step of normalization to ensure the design is normalized to the 5NF level. The product is a design where all attributes are atomic, every non key is fully dependent on the full key, there exist no transitive dependencies, and there exist no overlapping keys. In other words, the consistency of the data in the database can be logically guaranteed.

**Last Minute Advice**

*1) New Data From Bob*

The first dataset given by bob would cause difficulties due to the current design not directly relating the retailer (RetID) to a particular order. In the current implementation, the relationship of an order to a retailer is derived through either the EmpID of the sales person associated with the order or the BlendED associated with the coffee purchased by the customer. The metadata of the `Orders` table would have to be altered in order to accommodate the direct relation of an order to a store, without defining a sales person or a coffee purchase.

The second dataset would simply require the updating of the NumLB in the tuple in Orders with primary key OrderID = 2055 from 5 to 1. No metadata changes are required in this case.

*2) Information from Sam*

On its own, recording a new store in New York named “New York 2” would only require one insertion into the database. A retail id (RetID) would have to be generated for the retailer. If this opening were to include another manager, employees, or blends - those would have to be added separately but they too will not be difficult to accommodate. No metadata changes are required in this case.

Updating the price of the Kona coffee beans would take multiple updates. One update would update the price of the bean itself, in the beans table. The second would require the updating of the PricePerLB of all blends containing that bean - which is currently 5 blends. While this may seem like a lot of effort, the computational effort of updating all blends that contain an updated bean will still be far less than the computational effort required to calculate the PricePerLB of a blend whenever someone queries it’s price. No metadata changes are required in this case either.

*3) Data items from Bob that “look different”*

The first data item is different in that it does not contain the #lb field that data items usually contain. This difference is negligible in regards to the recording (insertion) of the data into the database. Because NumLB is a non-key attribute and we did not explicitly define not-null in our metadata for the attribute, we will simply have a missing value.

The second data item is not a problem, in regards to the missing Qty field, for the same reason as inserting the missing NumLB was not a problem. However, the real challenge is what to do about the fact that although we already have an OrderID 3016 associated with CustID AB101, we are now told that OrderID 3016 is actually associated with SS205. This would be a business logic question that should be asked - “Can two customers share an order”. If customers can not - either the new data or the old data is incorrect. In its current implementation, the database would require the updating of OrderID 3016 to be associated with SS205 instead of AB101 as well as the updating of the new Shipping Cost to $16.00 or we would simply not insert the data. Either way, we need to ask which record is actually correct.

*4) Number of orders field*

The number of orders paced for a customer or by an employee can be calculated easily, without a field ‘number of orders placed’ by running a query with the COUNT() operator on the Orders table where either the CustID or the EmpID is related to the person you are interested in.

**Conclusion**

A successful database architecture for order and inventory management requires, at highest priority, data consistency and availability. The relational database model specializes in both fields and offers with it the support of enterprise level software like SQL server.

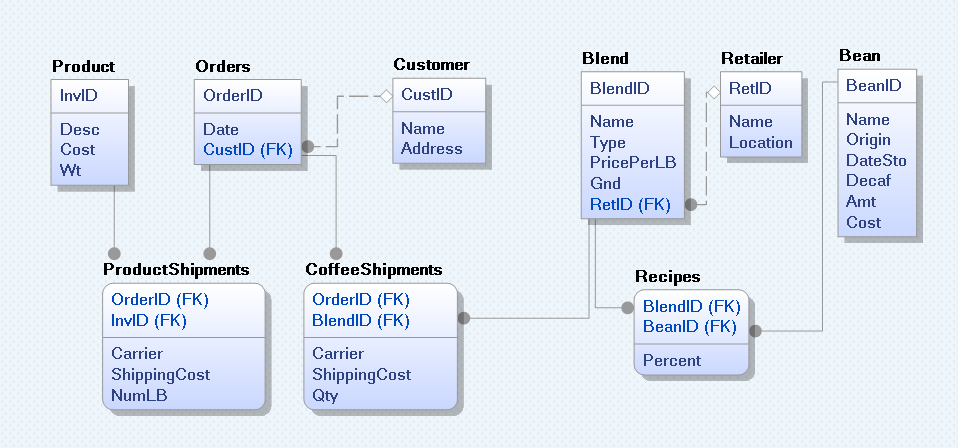
The solution presented in this report examined thoroughly the CBC project requirements and the sample data in order to generate a database architecture that supports all of the required features while at the same time ensuring data consistency from a theoretical basis, rooted in the relational database design presented.

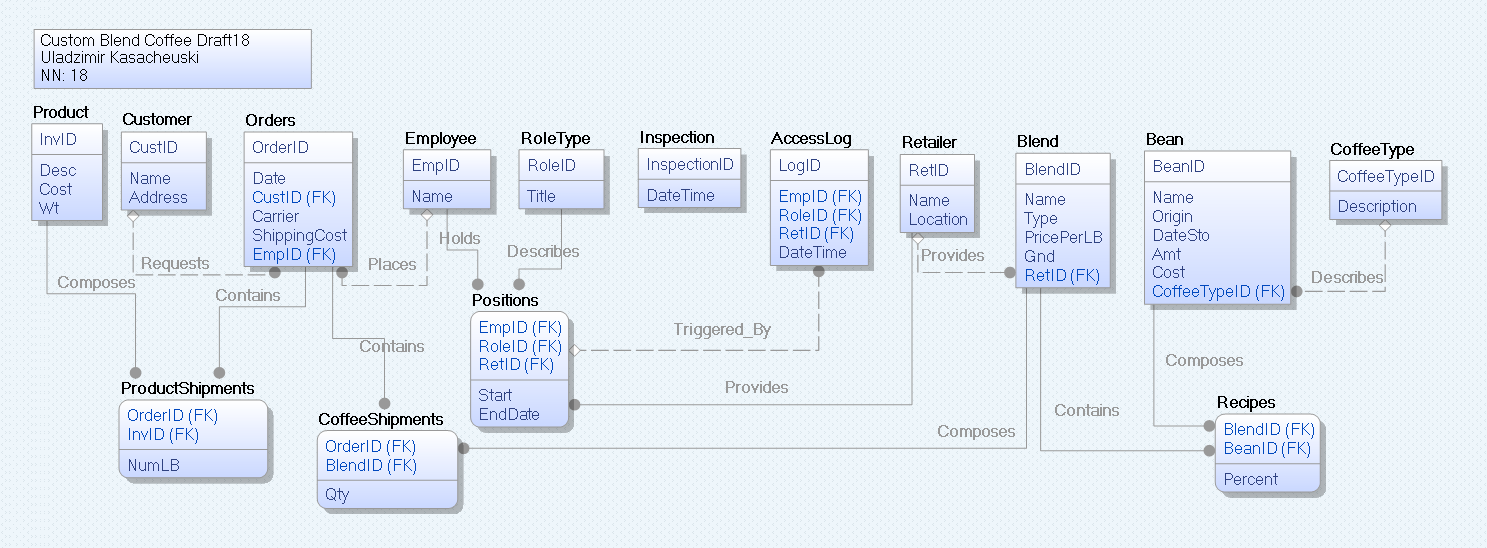
**Appendices**

*A1 - Historical Progression of the LDM*

Progression from the first LDM to the second dealt solely with the accommodation of the increased scope communicated in Memo 4. This involved the addition of tracking employees, discerning between employee roles, and tracking which sales person placed which order. The final adjustment from the second LDM to the current model dealt with decreasing the database complexity, as the second LDM implemented features that were not required by the scope but which complicated the usage and maintenance of the database.

Following are the first and second LDM.





*A2 - DDL for Prototype Creation*

CREATE TABLE Employee

(

EmpID char(7) NOT NULL ,

Name char(30) NULL ,

PRIMARY KEY CLUSTERED (EmpID ASC)

)

go

CREATE TABLE Retailer

(

RetID char(4) NOT NULL ,

Name char(30) NULL ,

Location char(30) NULL ,

PRIMARY KEY CLUSTERED (RetID ASC)

)

go

CREATE TABLE SalesPerson

(

EmpID char(7) NOT NULL ,

RetID char(4) NULL ,

StartDate date NULL ,

EndDate date NULL ,

PRIMARY KEY CLUSTERED (EmpID ASC),

FOREIGN KEY (EmpID) REFERENCES Employee(EmpID),

FOREIGN KEY (RetID) REFERENCES Retailer(RetID)

)

go

CREATE TABLE Blend

(

BlendID char(4) NOT NULL ,

Name char(30) NULL ,

Type char(1) NULL ,

PricePerLB smallmoney NULL ,

Gnd char(1) NULL ,

RetID char(4) NULL ,

PRIMARY KEY CLUSTERED (BlendID ASC),

FOREIGN KEY (RetID) REFERENCES Retailer(RetID)

)

go

CREATE TABLE CoffeeType

(

CoffeeTypeID int NOT NULL ,

Description char(7) NULL ,

PRIMARY KEY CLUSTERED (CoffeeTypeID ASC)

)

go

CREATE TABLE Bean

(

BeanID char(4) NOT NULL ,

Name char(30) NULL ,

Origin char(30) NULL ,

DateSto date NULL ,

Amt int NULL ,

Cost smallmoney NULL ,

CoffeeTypeID int NULL ,

PRIMARY KEY CLUSTERED (BeanID ASC),

FOREIGN KEY (CoffeeTypeID) REFERENCES CoffeeType(CoffeeTypeID)

)

go

CREATE TABLE Recipes

(

BlendID char(4) NOT NULL ,

BeanID char(4) NOT NULL ,

Percentage int NULL ,

PRIMARY KEY CLUSTERED (BlendID ASC,BeanID ASC),

FOREIGN KEY (BlendID) REFERENCES Blend(BlendID),

FOREIGN KEY (BeanID) REFERENCES Bean(BeanID)

)

go

CREATE TABLE Customer

(

CustID char(5) NOT NULL ,

Name char(30) NULL ,

Address char(30) NULL ,

PRIMARY KEY CLUSTERED (CustID ASC)

)

go

CREATE TABLE Orders

(

OrderID int NOT NULL ,

Date date NULL ,

CustID char(5) NULL ,

Carrier char(5) NULL ,

ShippingCost smallmoney NULL ,

EmpID char(7) NULL ,

PRIMARY KEY CLUSTERED (OrderID ASC),

FOREIGN KEY (CustID) REFERENCES Customer(CustID),

FOREIGN KEY (EmpID) REFERENCES Employee(EmpID)

)

go

CREATE TABLE Product

(

InvID int NOT NULL ,

Description char(30) NULL ,

Cost smallmoney NULL ,

Wt int NULL ,

PRIMARY KEY CLUSTERED (InvID ASC)

)

go

CREATE TABLE ProductShipments

(

OrderID int NOT NULL ,

InvID int NOT NULL ,

Qty int NULL ,

PRIMARY KEY CLUSTERED (OrderID ASC,InvID ASC),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (InvID) REFERENCES Product(InvID)

)

go

CREATE TABLE Manager

(

EmpID char(7) NOT NULL ,

RetID char(4) NULL ,

StartDate date NULL ,

EndDate date NULL ,

LastAccess datetime NULL ,

PRIMARY KEY CLUSTERED (EmpID ASC),

FOREIGN KEY (EmpID) REFERENCES Employee(EmpID),

FOREIGN KEY (RetID) REFERENCES Retailer(RetID)

)

go

CREATE TABLE CoffeeShipments

(

OrderID int NOT NULL ,

BlendID char(4) NOT NULL ,

NumLB int NULL ,

PRIMARY KEY CLUSTERED (OrderID ASC,BlendID ASC),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (BlendID) REFERENCES Blend(BlendID)

)

go

CREATE TABLE Bureaucrat

(

EmpID char(7) NOT NULL ,

StartDate date NULL ,

EndDate date NULL ,

NextInspection datetime NULL ,

PRIMARY KEY CLUSTERED (EmpID ASC),

FOREIGN KEY (EmpID) REFERENCES Employee(EmpID)

)

go

*A3 - DML for Data Load*

INSERT INTO Employee (EmpID,Name) VALUES ('EMP108','Bill Brown');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP109','Jane Smith');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP202','Frank Martin');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP213','Anne Dough');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP303','Mike Green');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP309','Alice Grey');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP400','Bill Brown');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP0000','Bob Bureaucrat');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP0101','Sam Supervison');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP0102','Mary Manager');

INSERT INTO Employee (EmpID,Name) VALUES ('EMP0103','Fred Foreman');

INSERT INTO Retailer (RetID,Name,Location) VALUES ('CBC1','CBC Store','Indy, IN');

INSERT INTO Retailer (RetID,Name,Location) VALUES ('NYC1','New York 1','NYC, NY');

INSERT INTO Retailer (RetID,Name,Location) VALUES ('SFO3','SFO store','SFO, CA');

INSERT INTO SalesPerson (EmpID,RetID,StartDate,EndDate) VALUES ('EMP108','CBC1','2016-06-03',NULL);

INSERT INTO SalesPerson (EmpID,RetID,StartDate,EndDate) VALUES ('EMP109','NYC1','2016-06-03',NULL);

INSERT INTO SalesPerson (EmpID,RetID,StartDate,EndDate) VALUES ('EMP202','NYC1','2016-06-05',NULL);

INSERT INTO SalesPerson (EmpID,RetID,StartDate,EndDate) VALUES ('EMP213','CBC1','2016-05-10','2016-09-20');

INSERT INTO SalesPerson (EmpID,RetID,StartDate,EndDate) VALUES ('EMP303','SFO3','2016-06-10',NULL);

INSERT INTO SalesPerson (EmpID,RetID,StartDate,EndDate) VALUES ('EMP309','SFO3','2016-06-10',NULL);

INSERT INTO SalesPerson (EmpID,RetID,StartDate,EndDate) VALUES ('EMP400','SFO3','2016-07-10',NULL);

INSERT INTO Blend (BlendID,Name,Type,PricePerLB,Gnd,RetID) VALUES ('CR01','Kona #1','R','80.5','G','CBC1');

INSERT INTO Blend (BlendID,Name,Type,PricePerLB,Gnd,RetID) VALUES ('CR02','Kona #2','R','81.5','G','CBC1');

INSERT INTO Blend (BlendID,Name,Type,PricePerLB,Gnd,RetID) VALUES ('CS03','Dark','S','49.5','G','CBC1');

INSERT INTO Blend (BlendID,Name,Type,PricePerLB,Gnd,RetID) VALUES ('N02','Spec02','S','24.7','B','NYC1');

INSERT INTO Blend (BlendID,Name,Type,PricePerLB,Gnd,RetID) VALUES ('CS05','House','S','17.0','G','CBC1');

INSERT INTO Blend (BlendID,Name,Type,PricePerLB,Gnd,RetID) VALUES ('SFOP','Blend P','R','13.1','G','SFO3');

INSERT INTO Blend (BlendID,Name,Type,PricePerLB,Gnd,RetID) VALUES ('CS04','SoAm','S','16.68','B','CBC1');

INSERT INTO Blend (BlendID,Name,Type,PricePerLB,Gnd,RetID) VALUES ('SFOA','Blend A','S','12.88','B','SFO3');

INSERT INTO CoffeeType (CoffeeTypeID,Description) VALUES ('1','Decaf');

INSERT INTO CoffeeType (CoffeeTypeID,Description) VALUES ('0','Regular');

INSERT INTO Bean (BeanID,Name,Origin,DateSto,Amt,Cost,CoffeeTypeID) VALUES ('K001','Kona #1','Kona, HI','May 2016','225','80.5','0');

INSERT INTO Bean (BeanID,Name,Origin,DateSto,Amt,Cost,CoffeeTypeID) VALUES ('K002','Kona #2','Kona, HI','May 2016','110','81.5','1');

INSERT INTO Bean (BeanID,Name,Origin,DateSto,Amt,Cost,CoffeeTypeID) VALUES ('B234','Brazil Dark','Santos, BR','Apr 2016','225','18.5','0');

INSERT INTO Bean (BeanID,Name,Origin,DateSto,Amt,Cost,CoffeeTypeID) VALUES ('CO23','Supremo','Columbia','Jun 2016','50','12.65','0');

INSERT INTO Bean (BeanID,Name,Origin,DateSto,Amt,Cost,CoffeeTypeID) VALUES ('P002','High Grown','Peru D','Jul 2016','100','13.1','1');

INSERT INTO Bean (BeanID,Name,Origin,DateSto,Amt,Cost,CoffeeTypeID) VALUES ('CR05','Tarrazu','Costa Rico','Jan 2015','33','14.8','0');

INSERT INTO Bean (BeanID,Name,Origin,DateSto,Amt,Cost,CoffeeTypeID) VALUES ('P001','High Grown','Peru','Jul 2016','100','13.1','0');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('CR01','K001','100');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('CR02','K002','100');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('CS03','B234','50');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('CS03','K001','50');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('N02','B234','90');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('N02','K001','10');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('CS05','B234','75');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('CS05','CO23','25');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('SFOP','P002','100');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('CS04','B234','60');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('CS04','CR05','20');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('CS04','P001','20');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('SFOA','CO23','50');

INSERT INTO Recipes (BlendID,BeanID,Percentage) VALUES ('SFOA','P001','50');

INSERT INTO Customer (CustID,Name,Address) VALUES ('AB101','John Doe','Indy');

INSERT INTO Customer (CustID,Name,Address) VALUES ('AC101','Anne Green','Indy');

INSERT INTO Customer (CustID,Name,Address) VALUES ('CC201','Bill Green','NYC');

INSERT INTO Customer (CustID,Name,Address) VALUES ('SS205','John Doe','SFO');

INSERT INTO Customer (CustID,Name,Address) VALUES ('LL101','Amy White','LA');

INSERT INTO Customer (CustID,Name,Address) VALUES ('QQ202','Gail Black','PHX');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('1012','2016-06-15','AB101','FedEx','4','EMP108');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('1013','2016-06-15','AC101','FedEx','8','EMP108');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('1015','2016-07-20','AC101','FedEx','4','EMP213');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('2000','2016-07-21','CC201','UPS','12','EMP109');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('2050','2016-08-01','SS205','UPS','16','EMP202');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('2051','2016-08-02','LL101','UPS','20','EMP108');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('2055','2016-08-10','QQ202','UPS','20','EMP303');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('3015','2016-08-10','AB101','UPS','12','EMP108');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('3016','2016-08-10','AB101','UPS','12','EMP309');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('3017','2016-08-10','LL101','FedEx','24','EMP400');

INSERT INTO Orders (OrderID,Date,CustID,Carrier,ShippingCost,EmpID) VALUES ('4000','2016-09-02','LL101','FedEx','32','EMP213');

INSERT INTO Product (InvID,Description,Cost,Wt) VALUES ('1345','CBC cup (W)','2.5','1');

INSERT INTO Product (InvID,Description,Cost,Wt) VALUES ('2208','CBC grinder','22.5','2');

INSERT INTO Product (InvID,Description,Cost,Wt) VALUES ('1346','CBC cup (B)','2.5','1');

INSERT INTO Product (InvID,Description,Cost,Wt) VALUES ('2209','AAA grinder','27.5','2');

INSERT INTO ProductShipments (OrderID,InvID,Qty) VALUES ('1013','1345','2');

INSERT INTO ProductShipments (OrderID,InvID,Qty) VALUES ('2000','2208','1');

INSERT INTO ProductShipments (OrderID,InvID,Qty) VALUES ('2050','1345','1');

INSERT INTO ProductShipments (OrderID,InvID,Qty) VALUES ('2050','1346','1');

INSERT INTO ProductShipments (OrderID,InvID,Qty) VALUES ('3015','2209','1');

INSERT INTO ProductShipments (OrderID,InvID,Qty) VALUES ('4000','2209','1');

INSERT INTO Manager (EmpID,RetID,StartDate,EndDate,LastAccess) VALUES ('EMP0101','CBC1','2016-06-03',NULL,NULL);

INSERT INTO Manager (EmpID,RetID,StartDate,EndDate,LastAccess) VALUES ('EMP0102','NYC1','2016-06-03',NULL,NULL);

INSERT INTO Manager (EmpID,RetID,StartDate,EndDate,LastAccess) VALUES ('EMP0103','SFO3','2016-06-03',NULL,NULL);

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('1012','CR01','1');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('1013','CR02','1');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('1015','CS03','1');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('2000','N02','1');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('2050','N02','2');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('2051','CS05','5');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('2055','SFOP','5');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('3015','CS04','1');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('3016','SFOA','1');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('3017','SFOP','4');

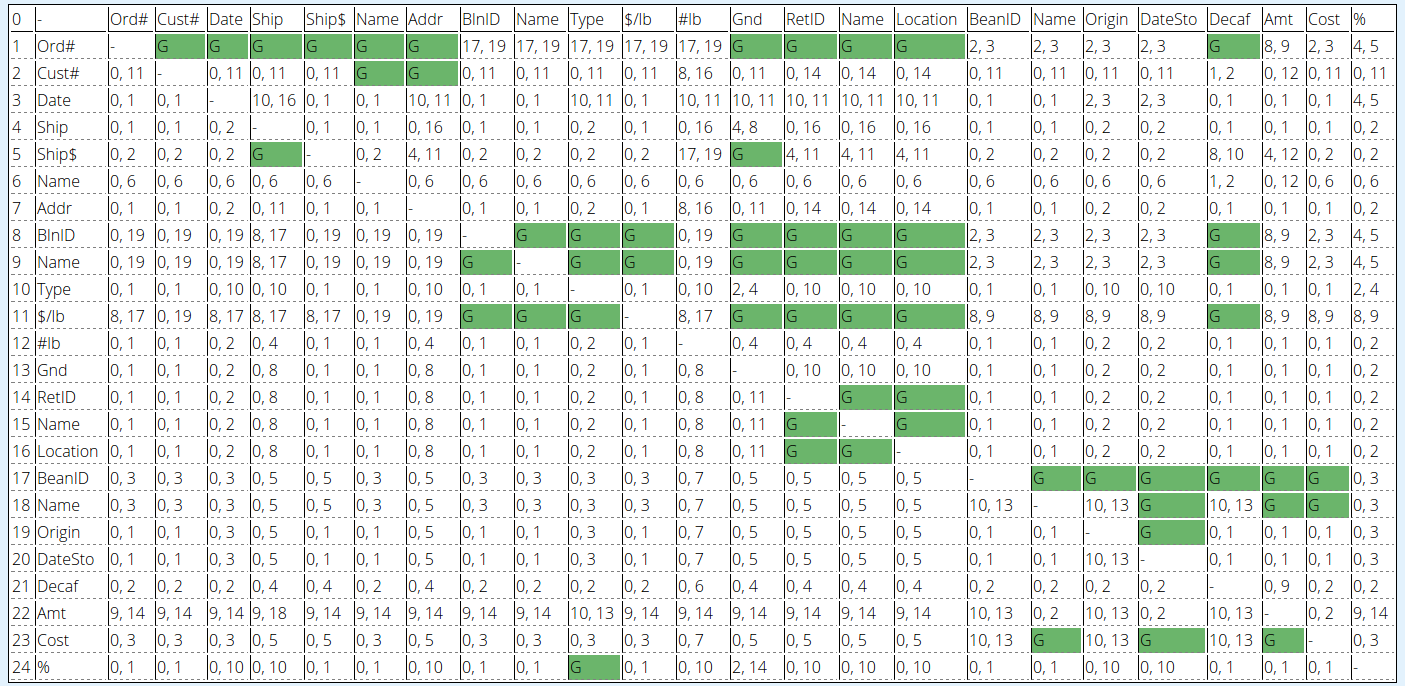
INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('4000','CS05','4');

INSERT INTO CoffeeShipments (OrderID,BlendID,NumLB) VALUES ('4000','CR01','2');

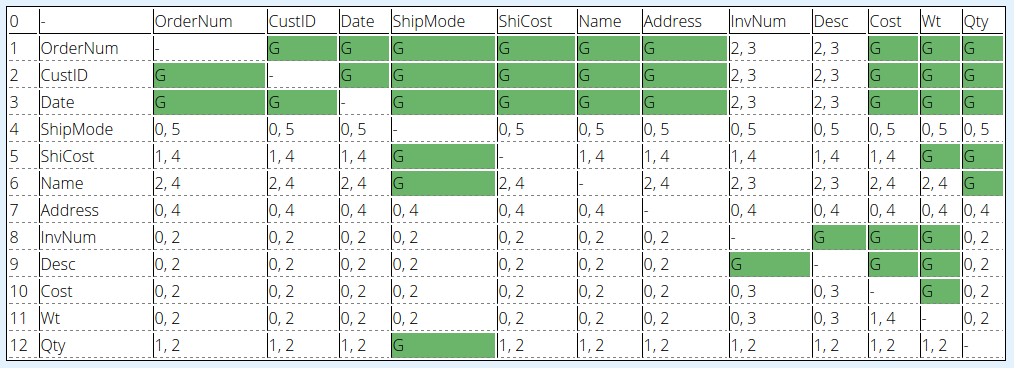
INSERT INTO Bureaucrat (EmpID,StartDate,EndDate,NextInspection) VALUES ('EMP0000',NULL,NULL,NULL);

*A4 - Functional Dependency Analysis*

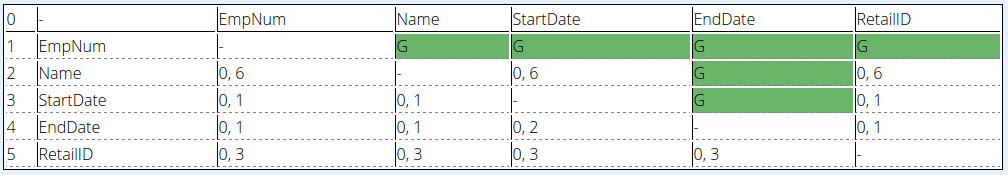
Coffee



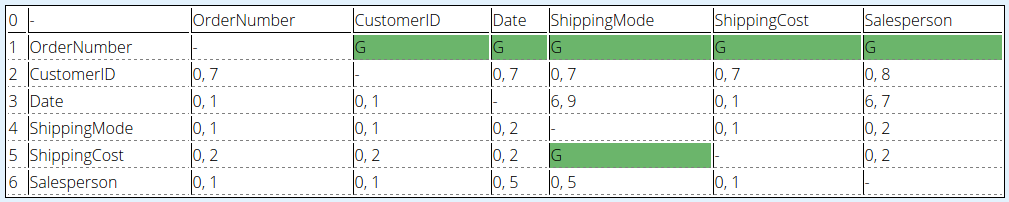
Inventory



SalesPersons



Orders By Salespersons



*FD’s that can be trivially denied on denied by business rules are not displayed below*

*Coffee Purchase FD’*

Ord# -> Cust#

Ord# -> Date

Ord# -> Ship

Ord# -> Ship$

Ord# -> Name

Ord# -> Addr

Ord# -> BlnID

Ord# -> Name

Ord# -> Type

Ord# -> $/lb

Ord# -> #lb

Ord# -> Gnd

Ord# -> RetID

Ord# -> Name

Ord# -> Location

Cust# -> Name

Cust# -> Addr

Ship$ -> #lb

BlnID -> Name

BlnID -> Type

BlnID -> $/lb

BlnID -> Gnd

BlnID -> RetID

BlnID -> Name

BlnID -> Location

Name -> BlnID

Name -> Type

Name -> $/lb

Name -> Gnd

Name -> RetID

Name -> Name

Name -> Location

RetID -> Name

RetID ->Location

Name -> RetID

Name -> Location

Location -> RetID

Location -> Name

BeanID -> Name

BeanID -> Origin

BeanID -> DateSto

BeanID -> Decaf

BeanID -> Amt

BeanID -> Cost

Name -> DateSto

Name -> Amt

Name -> Cost

{BlnID, BeanID} -> %

*Inventory Purchases FD’s*

OrderNum -> CustID

OrderNum -> Date

OrderNum -> ShipMode

OrderNum -> ShiCost

OrderNum -> Name

OrderNum -> Address

OrderNum -> Cost

OrderNum -> Wt

OrderNum -> Qty

CustID -> OrderNum

CustID -> Date

CustID -> ShipMode

CustID -> ShiCost

CustID -> Name

CustID -> Address

CustID -> Cost

CustID -> Wt

CustID -> Qty

Date -> Cost

Date -> Wt

InvNum -> Desc

InvNum -> Cost

InvNum -> Wt

Desc -> InvNum

Desc -> Cost

Desc -> Wt

*SalesPersons FD’s*

EmpNum -> Name

EmpNum -> StartDate

EmpNum -> EndDate

EmpNum -> RetailID

*Orders By SalesPersons FD’s*

OrderNumber -> CustomerID

OrderNumber -> Date

OrderNumber -> ShippingMode

OrderNumber -> ShippingCost

OrderNumber -> SalesPerson